Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчёт  
по лабораторной работе №2

«Работа с *JavaScript*»

|  |  |
| --- | --- |
| Проверил | Выполнил |
| Гончаревич А. Л. | ст. гр. №820602 |
|  | Д. С. Жуковский |

Минск 2021

**СОДЕРЖАНИЕ**

Введение 3

1 Цель работы 4

2 Теоретическая часть 5

2.1 Использование *JavaScript* на веб-страницах 5

2.2 Базовые понятия и конструкции *JavaScript* 6

3 Практическая часть 9

3.1 Описание используемых программных средств 9

3.2 Описание назначений фреймов 9

3.3 Руководство пользователю 13

Заключение 14

Список использованных источников 15

# Введение

*JavaScript* был представлен в 1995 году как способ добавления программ на веб-страницы в браузере *Netscape Navigator*. С тех пор этот язык был принят другими основными браузерами. Это сделало возможными современные веб-приложения – приложения, с которыми можно взаимодействовать напрямую, не перезагружая страницу для каждого действия. *JavaScript* также используется на более традиционных веб-сайтах для обеспечения различных форм интерактивности [1].

Важно отметить, что *JavaScript* почти не имеет ничего общего с языком программирования *Java*. Изначально первый имел название *Mocha*. Брендан Эйх создал его в 1992 году. В это же время компания *Sun* *Microsystems* заканчивала разработку *Java*. Эйх договорился с ними – и *Mocha* стал скриптовым компаньоном языка *Java*. Позднее *Mocha* был переименован в *LiveScript*, что также не принесло языку славы. Тогда, воспользовавшись популярностью своего напарника, разработчики дали ему название *JavaScript*.

*JavaScript* используют в основном в веб-разработке. Вместе с *HTML* и *CSS*, *JavaScript* создаёт идеальный набор для фронтенд-разработчика.

В веб-разработке *JavaScript* помогает обрабатывать любое действие пользователя – от заполнения формы до прокрутки страницы. В браузере срабатывает событие, которое запускает работу *JavaScript*-кода, только после этого на странице происходит изменение. На каждом сайте используется несколько скриптов, которые собирают данные посетителей, анализируют метрики и позволяют зарегистрировать аккаунт.

*JavaScript* используют также для следующих целей:

– разработка нативных приложений;

– бэкенд-разработка;

– создание ПО для техники;

– разработка десктопных приложений;

– создание расширений для браузера.

*JavaScript* – постоянно развивающийся язык с продуманной инфраструктурой и большой экосистемой. У этого языка нет конкурентов в его основной сфере применения, он справляется со всеми поставленными задачами. Поэтому сейчас нет необходимости в создании альтернативного варианта – он достаточно быстр, удобен и универсален.

Также для этого языка существуют фреймворки, в которых есть предварительно написанный код для использования в функциях и задачах программирования. Наиболее популярны *React*, *Angular*, *Vue*.

# Цель работы

Познакомиться с жаваскрипт.

Сделать задания с использованием жс.

# Теоретическая часть

## Использование *JavaScript* на веб-страницах

*JavaScript*-код может встраиваться в *HTML*-документы следующими способами:

– встроенные сценарии между парой тегов <*script*> и </*script*>;

– из внешнего файла, заданного атрибутом *src* элемента *script*;

– в обработчик события, заданный в качестве значения *HTML*-атрибута;

– как тело *URL*-адреса.

*HTML*-атрибуты обработчиков событий и адреса *URL* с псевдопротоколом *javascript* редко используются в современной практике программирования на языке *JavaScript* (они были более распространены на раннем этапе развития Интернета).

Встроенные сценарии также стали реже использоваться по сравнению с прошлым. Согласно философии программирования, известной как ненавязчивый *JavaScript* (*unobtrusive* *JavaScript*), содержимое (разметка *HTML*) и поведение (программный код на языке *JavaScript*) должны быть максимально отделены друг от друга. Следуя этой философии программирования, сценарии на языке *JavaScript* лучше встраивать в *HTML*-документы с помощью элементов *script*, имеющих атрибут *src*. Элемент *script* с атрибутом *src* ведёт себя точно так, как если бы содержимое указанного файла *JavaScript*-кода находилось непосредственно между тегами <*script*> и </*script*>.

При использовании атрибута *src* любое содержимое между открывающим и закрывающим тегами <*script*> игнорируется. При желании в качестве содержимого можно вставлять описание включаемого программного кода или информацию об авторском праве. Однако следует заметить, что инструменты проверки соответствия разметки требованиям стандарта *HTML*5 будут выдавать предупреждения, если между тегами <*script*> и </*script*> с атрибутом *src* будет находиться какой-либо текст, не являющийся пробельными символами или комментариями на языке *JavaScript*.

Использование тега *script* с атрибутом *src* даёт ряд преимуществ:

1. *HTML*-файлы становятся проще, так как из них можно убрать большие блоки *JavaScript*-кода. Это помогает отделить содержимое от поведения.
2. *JavaScript*-функцию или другой код, используемый несколькими *HTML*-документами, можно хранить в одном файле и считывать при необходимости. Это уменьшает объём занимаемой дисковой памяти и намного облегчает поддержку программного кода, так как отпадает необходимость править каждый файл при изменении кода.
3. Если сценарий на языке *JavaScript* используется сразу несколькими страницами, он будет загружаться браузером только один раз, при первом его использовании. Последующие страницы будут извлекать его из кэша браузера.
4. Атрибут *src* принимает в качестве значения произвольный *URL*-адрес, поэтому *JavaScript*-программа или веб-страница с одного веб-сервера может воспользоваться кодом (например, из библиотеки подпрограмм), предоставляемым другими веб-серверами. Многие рекламодатели в Интернете используют этот факт.

## Базовые понятия и конструкции *JavaScript*

### Переменная – это контейнер для таких значений, как числа, используемые в арифметических выражениях, строки, которые можно использовать как части предложений, или объекты, имеющие множество ассоциируемых с ними свойств. Особенностью переменных является то, что их состояние может изменяться. Чтобы использовать переменную, сначала необходимо её объявить. Для этого существует ключевое слово *var*, после которого указывается имя переменной. Как только переменная объявлена, ей можно присвоить значение. Для этого указывается имя переменной, затем следует знак равенства, а за ним значение, которое необходимо присвоить. Разрешено объявлять переменную и задавать ей значение одновременно [2].

### Массив – это один объект, который содержит значения, заключённые в квадратные скобки и разделённые запятыми. Отдельный массив, в свою очередь, является переменной. Для объявления такой переменной также используется ключевое слово *var*, но значением такой переменной является высокоуровневый спископодобный объект [2]. Размер массива может увеличиваться и уменьшаться в любое время, что говорит о динамичности данного вида переменных. Массивы в *JavaScript* индексируются с нуля: первый элемент массива имеет индекс, равный нулю, а индекс последнего элемента равен значению свойства массива *length* уменьшенному на единицу. Таким образом, чтобы, например, вывести значение элемента массива на консоль, или сохранить значение элемента в другой переменной, необходимо указать название массива и индекс данного элемента в квадратных скобках.

### В зависимости от результата вычисления некоторого логического выражения можно указать различные пути выполнения программы при помощи условных операторов *JavaScript*. Наиболее распространённым типом условного оператора является оператор *if else*. При использовании данного оператора сперва записывается ключевое слово *if*, после которого в круглых скобках указывается логическое выражение. Далее в фигурных скобках располагается поток действий, выполняемых, если результатом вычисления логического выражения является истина. В части *else* располагается поток действий, выполняемых при противоположном результате вычисления выражения. При отсутствии альтернативных действий последнюю часть можно опустить. Если требуется указать дополнительные варианты выполнения программы, то есть возможность расположить одну или несколько частей *else if* с дополнительными условиями между основными блоками *if* и *else*. Также существует оператор *switch*. Он принимает единственное выражение, которое указывается за соответствующим ключевым словом в круглых скобках, а затем просматривает ряд вариантов, представленных блоками, начинающимися с ключевого слова *case* со сравниваемыми выражениями, пока не будет найден вариант, соответствующий этому значению, после чего выполняет код, назначенный этому варианту. Есть возможность указать набор действий, выполняемых в случае, когда ни один из вариантов не соответствует входному выражению. Для этого используется ключевое слово *default*, но без указания выражения для сравнения со входным. Для того, чтобы не допустить так называемого сквозного выполнения операторов, требуется завершать каждый блок варианта ключевым словом *break*, если, бизнес-логика противоречит такому поведению. Всё тело оператора *switch* заключается в фигурные скобки.

### С помощью циклов можно выполнить набор действий несколько раз. Объявление оператора *for* в *JavaScript* происходит с указанием начальных значений итерационных переменных, а также объявлений этих переменных при необходимости, условия, при котором будут выполняться выражения тела цикла, а также оператора, изменяющего значения итерационных переменных. Тело цикла заключается в фигурные скобки.

### В общем случае, функция – это «подпрограмма», которую можно вызывать из внешнего (или внутреннего, в случае рекурсии) по отношению к функции кода. Как и сама программа, функция состоит из последовательности инструкций, называемой телом функции. Для объявления функции используется ключевое слово *function*, за которым следуют название функции и открытая и закрытая скобки. Тело функции оборачивается фигурными скобками. Для того, чтобы вернуть значение, достаточно указать ключевое слово *return* и возвращаемое выражение в конце её тела. Дополнительно указывать возвращаемый тип при объявлении функции не требуется. Также можно передать параметры, которые затем могут использоваться в выражениях в теле функции. Для этого следует записать параметры внутри круглых скобок, разделив их запятыми. Типы параметров указывать не требуется.

### Объектная модель документа (*DOM*) – это программный интерфейс для *HTML*-документов. *DOM* предоставляет структурированное представление документа и определяет то, как эта структура может быть доступна из программ, которые могут изменять содержимое, стиль и структуру документа. Представление *DOM* состоит из структурированной группы узлов и объектов, которые имеют свойства и методы. По существу, *DOM* соединяет веб-страницу с языками описания сценариев либо языками программирования [2]. Весь документ в целом, заголовок, таблицы внутри документа, заголовки таблицы, текст внутри ячеек таблицы – это части объектной документной модели для этого документа, поэтому все они могут быть доступны и могут изменяться с помощью *DOM* и *JavaScript*. Программирование *DOM* может быть чем-то простым, например, вывод сообщения с использованием функции *alert*() объекта *window*, или использование более сложных методов *DOM*, которые создают новое содержимое. *Document*, *window* – это объекты, чьи интерфейсы, как правило, часто используются в программировании *DOM*. Говоря простыми словами, объект *window* представляет что-то вроде браузера, а *document* – корень самого документа [3].

Наиболее распространены следующие члены программного интерфейса:

– *document*.*getElementById*(*id*);

– *document*.*getElementsByTagName*(*name*);

– *document*.*createElement*(*name*);

– *parentNode*.*appendChild*(*node*);

– *element*.*innerHTML*.

# Практическая часть

## Описание используемых программных средств

### *Visual* *Studio Code* – редактор исходного кода, разработанный *Microsoft* для *Windows*, *Linux* и *macOS*. Позиционируется как лёгкий редактор кода для кроссплатформенной разработки. Включает в себя отладчик, инструменты для работы с *Git*, подсветку синтаксиса, *IntelliSense* и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Также к установке доступны различные расширения. Например, чтобы не обновлять созданную *HTML*-страницу в браузере после каждого редактирования кода, можно установить расширение *Live Server*, и открыть страничку с помощью контекстного меню. Тогда страница будет автоматически обновляться после каждого изменения. *Visual Studio Code* распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

### *Notepad*++ – бесплатный редактор исходного кода, яляющийся альтернативой блокноту или любому другому редактору, поддерживающий множество языков. Работа в среде *MS Windows* регулируется Стандартной общественной лицензией *GNU*. Основанный на мощном компоненте редактирования *Scintilla*, *Notepad* ++ написан на *C* ++ и использует чистый *Win*32 *API* и *STL*, что обеспечивает более высокую скорость выполнения и меньший размер программы.

## Описание назначений фреймов

### Начальная страница служит для первоначального ознакомления пользователя с сайтом. В левой области расположены ссылки на страницы, на которых демонстрируется различный функционал. Открытая страница будет отображаться в правой области. Поскольку ещё никакая ссылка не была выбрана, в правой области располагается текст-заглушка. На данную начальную страницу можно попасть со страницы «Учёба» по гиперссылке «Прочее» и вернуться обратно с помощью кнопки «Вернуться». Данная страница приведена на рисунке 1.

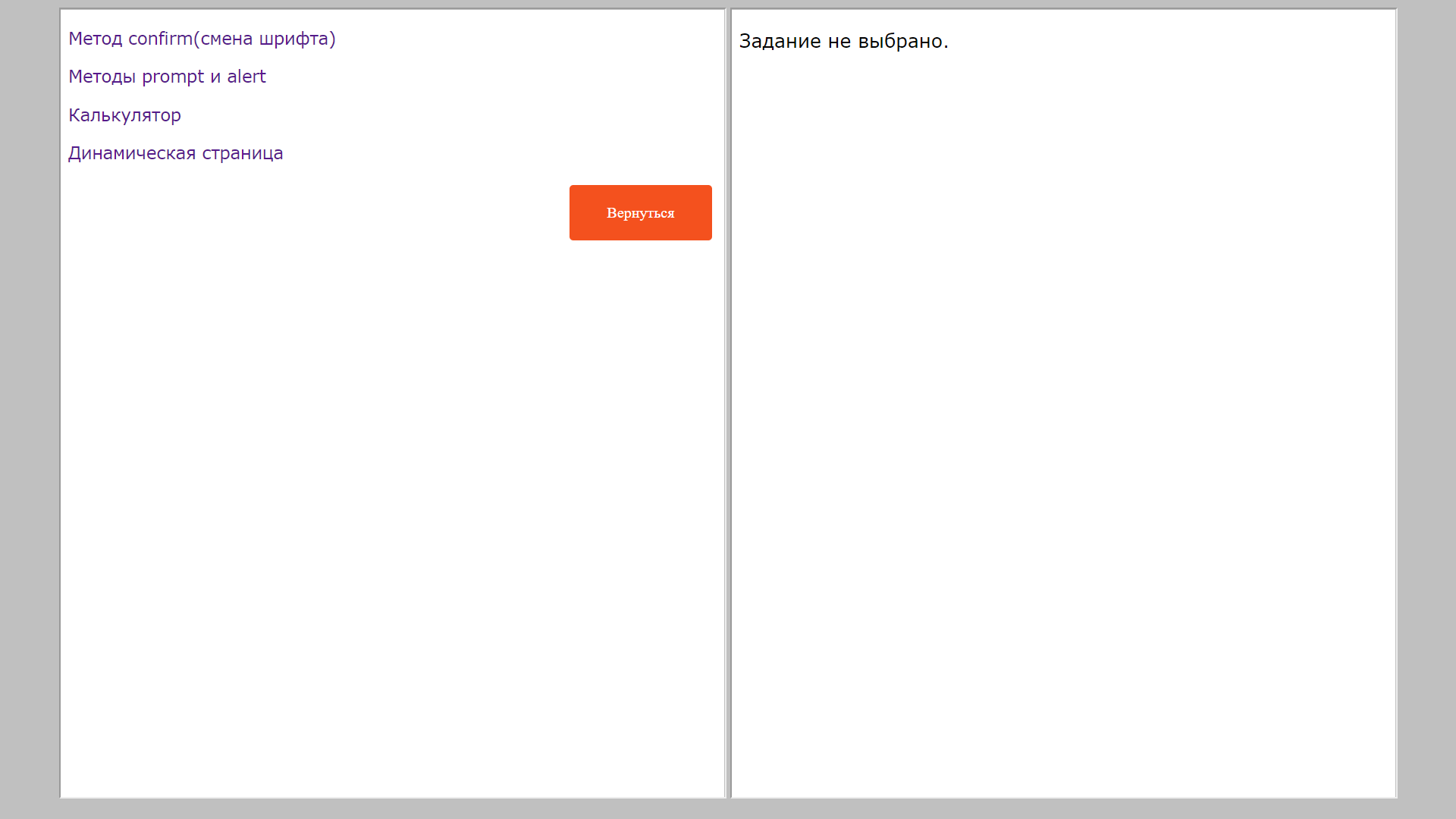


Рисунок 1 – Начальная страница

### Страница «Метод *confirm*(смена шрифта)» служит для демонстрации работы метода *confirm* и реализует смену шрифта данного фрейма со стандартного (*Times New Roman* в браузере *Google* *Chrome*) на шрифт *Arial* по нажатию кнопки. Страница изображена на рисунке 2.

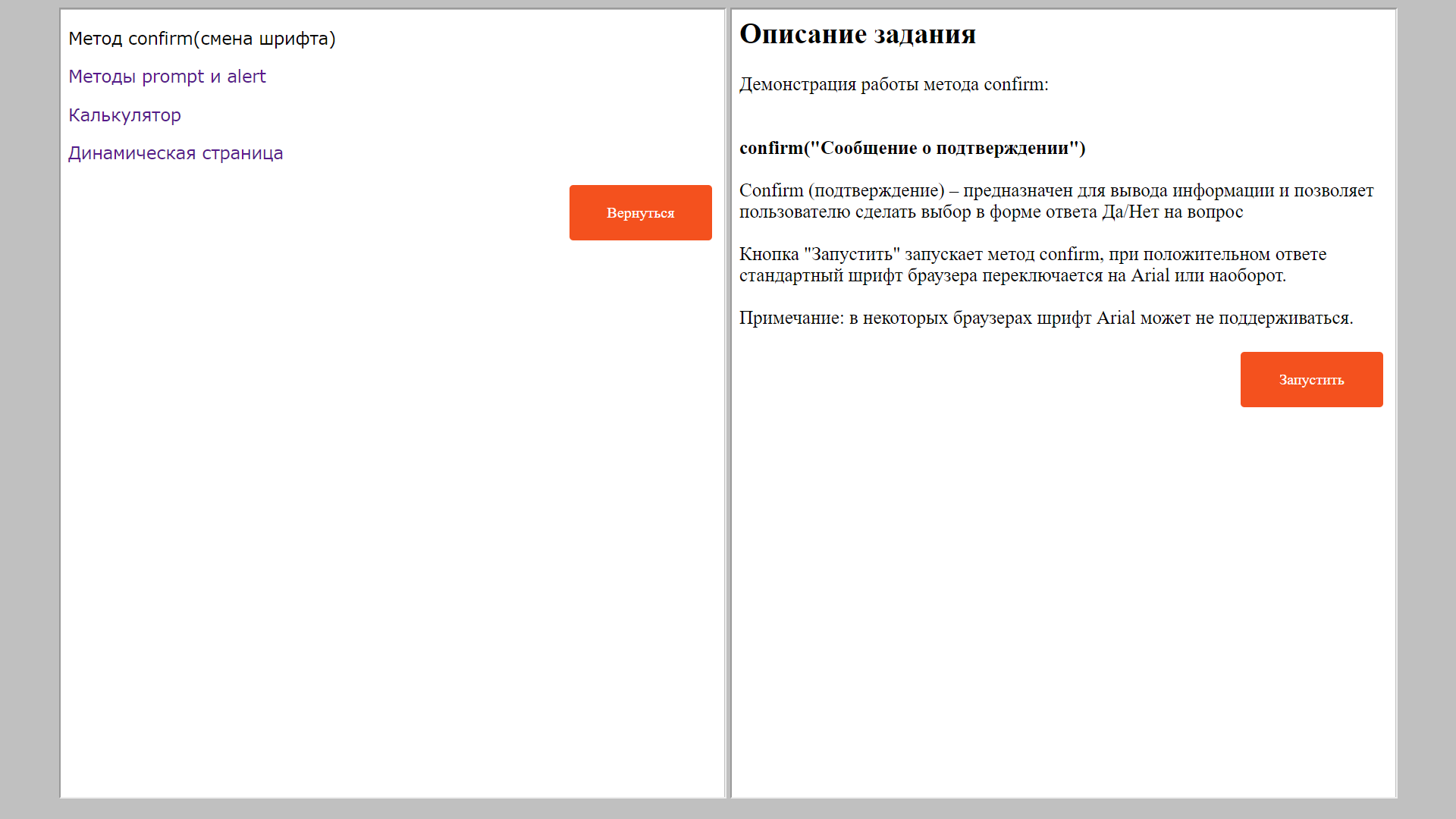


Рисунок 2 – Страница «Метод *confirm*(смена шрифта)»

### Страница «Метод *prompt* и *alert*» служит для демонстрации работы методов *prompt* и *alert* и реализует их работу посредством взаимодействия с пользователем. Данная страница приведена на рисунке 3

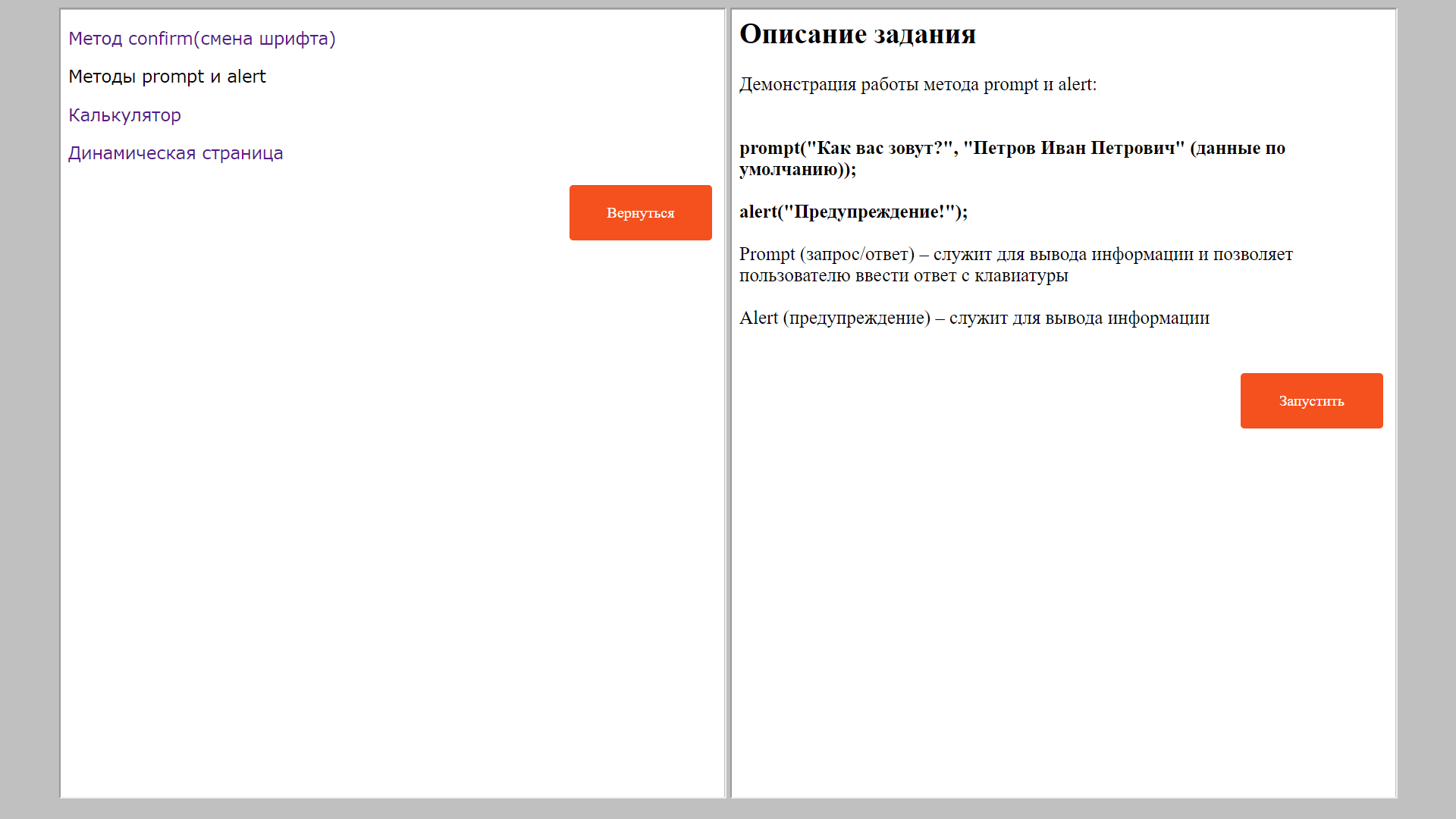


Рисунок 3 – Страница «Методы *prompt* и *alert*»

### Страница «Калькулятор» предоставляет пользователю достаточно большой функционал для вычисления различных выражений. Пользователь может складывать, вычитать, делить и умножить как обычные числа, так и десятичные. Также есть возможность вычислить инкремент или декремент текущего числа. Дополнительно присутствуют функции очистки поля для ввода, удаления последнего символа, смены знака и получения значения одного процента от текущего числа, то есть деление его на 100. Над полем для ввода выражений располагается строка, представляющая последнюю проделанную операцию и последнее полученное значение. Ввод производится только с помощью кнопок. Данная страница представлена на рисунке 4.

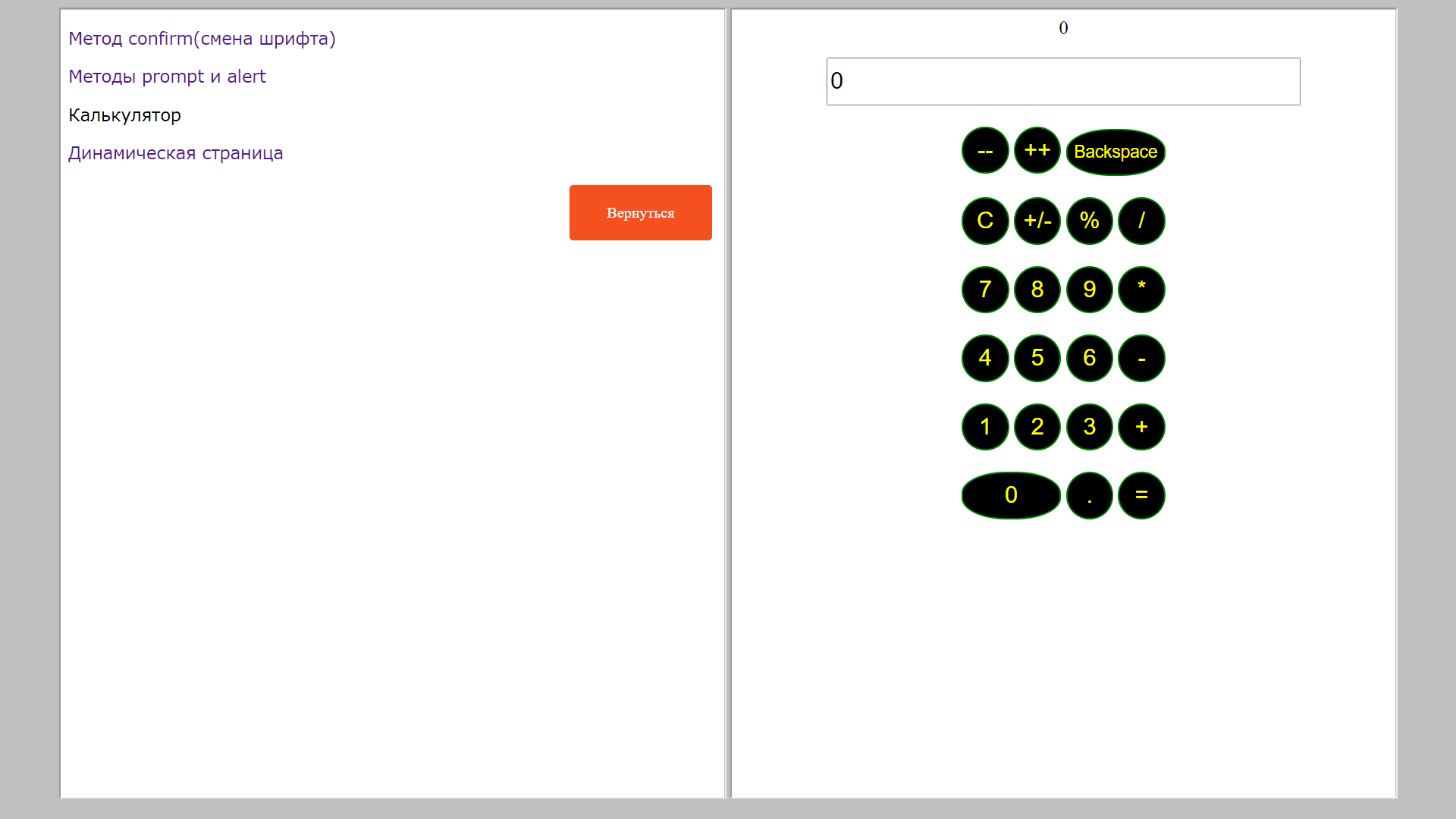


Рисунок 4 – Страница «Калькулятор»

### Страница «Динамическая страница» представляет собой таблицу, содержащую слова, которые можно перевести на один из трёх доступных языков: белорусский, английский, французский. Перевод производится посредством выбора переключателя. Данная страница приведена на рисунке 5.

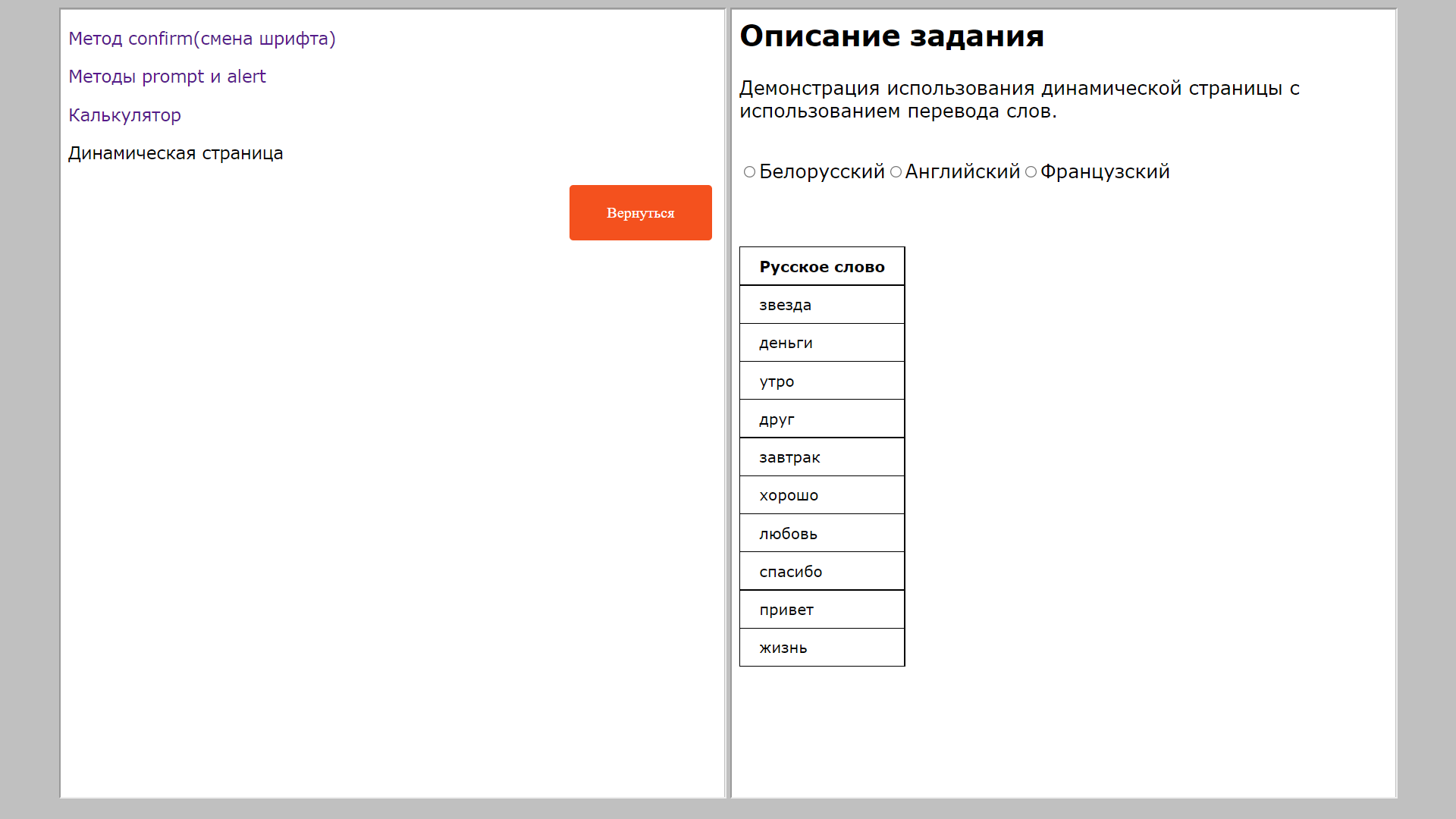


Рисунок 5 – Страница «Динамическая страница»

## Руководство пользователю

Со стартовой страницы можно попасть на одну из четырёх страниц. При выборе ссылки она окрасится в чёрный цвет, пока будет находиться в фокусе, а страница, содержащая функционал, будет загружена в область справа. С помощью кнопки можно вернуться на страницу «Учёба».

Если будет нажата первая ссылка, то загрузится страница с описанием задания, а также встроенного метода *confirm*. Шрифт страницы изменится, если пользователь после нажатия кнопки «Запустить» выберет утвердительный ответ, иначе ничего не произойдёт. После смены шрифта можно вернуть состояние страницы в исходное повторным вызовом метода и утвердительным ответом.

При выборе второй ссылки пользователь увидит в соседнем фрейме описание второго задания, а также встроенных методов *prompt* и *alert*. По нажатию кнопки появится всплывающее уведомление сверху с предложением ввести данные вместо данных по умолчанию. Если пользователь выберет вариант «Отмена», то ничего не произойдёт, иначе после ввода новой информации и выбора утвердительного ответа появится новое уведомление-приветствие со значением, введённом ранее.

Для пользования калькулятором следует нажимать кнопки, изменяющие свой цвет при наведении курсора на них. Текстовое поле предназначено только для отображения введённых символов, ввести в него ничего не получится, поскольку оно снабжено модификатором *readonly*. После вычисления значения выражения введённые операции и результат отобразятся над текстовым полем. При повторном вычислении выражения, то есть при нажатии кнопки равенства, пользователю отобразится последний вычисленный результат без исходной последовательности операций.

Последняя ссылка ведёт на динамическую страницу, где сперва отображается только одна колонка таблицы со словами-оригиналами. При выборе одного из переключателей динамически будет создаваться вторая колонка со словами, переведёнными на соответствующий язык. Если в данном состоянии снова изменить положение переключателя, то вторая колонка будет замещаться столбцом, содержащим переводы слов на другой язык.

# Заключение

# Список использованных источников

[1] Eloquent JavaScript / Marijn Haverbeke, 2018.

[2] https://developer.mozilla.org/

[3] JavaScript for .NET Developers / Packt Publishing Ltd -- Ovais Mehboob Ahmed Khan, 2016